Classifying Russian Propaganda Tweets from the 2016 Election with an LSTM Recurrent Neural Network

Javier Echevarría Cuesta, James Schull & Talal Rishani

javierec@stanford.edu
jschull@stanford.edu
trishani@stanford.edu

Motivation & Problem Definition

- During the 2016 elections, 677,775 people were exposed to Twitter posts from more than 50,000 automated accounts with links to the Russian government.
- **PROBLEM DEFINITION:** Given the text of a tweet from 2016, **predict whether or not it was from a Russian propaganda account.**

Approach

- **Data:** 3-million tweet training set from two sources: a 2018 NBC dataset containing 200,000 Russian troll tweets, and a Harvard dataset containing tweet ids of 280 million tweets related to the 2016 Presidential election.
- **Architecture:** LSTM whose unit outputs are each fed into a logistic regression unit; the resulting vector of activations is averaged to generate the final prediction.
- Tweets transformed into real-valued vector inputs using word embeddings that we trained on a vocabulary that we built ourselves.

Error Analysis & Discussion

- **The model learns frequent Russian hashtags and retweets**

"RT @NickAPappas: Trumpers, explain how, and more UNK WHERE Hillary will "rig" this election."

[0.2747, 0.9361, 0.9156, …….] —> 1: NickAPappas is an account frequently retweeted by troll accounts

- **Why is it falsely classifying some non-Russian tweets?**

"Hop on the Trump train my friends!!! This man can do great things for our country. MAGA!!! #Trump2016"

['0.4012', '0.6931', '0.6939', …….] —> 1: Some tweets sound very much like an inflammatory pro-Trump troll account…because they're very pro-Trump!

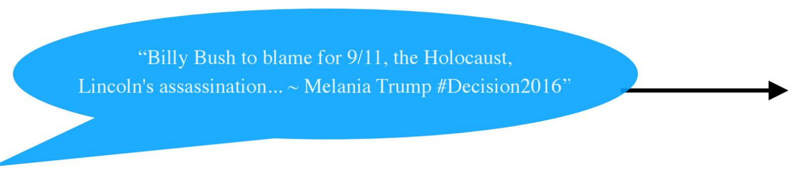
- **Why is it missing some Russian tweets?**

"RT UNK Trump is NOT a racist! UNK"

['0.2747', '0.2137', '0.0367', '0.0285', '0.056', '0.0351', '0.1654', '0.1778'] —> 0: Lots of unknown words, difficult to learn logical relations with variation in spelling ("not" versus "NOT")

Future

- With more time, we would train the model on data from different time periods, in order to see if it could generalize to detect suspicious accounts in real-time.



Results

Model	Train Accuracy	Test Accuracy
Logistic Regression (w/ one-hot vectors)	63.2%	54.8%
LSTM	94.3%	89.8%

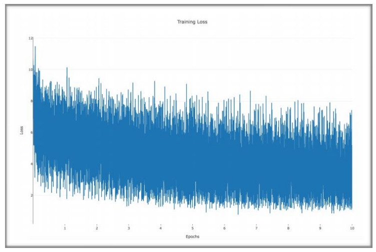


Fig 1. Training loss

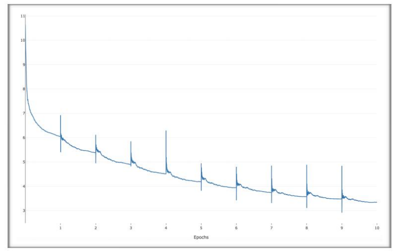


Fig 2. Mean batch loss per epoch

Architecture